#15

PATENT
Atty. Docket: 2207/5915

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Prudvi, et. al.		
Serial No.:	09/212,291	Examiner:	T. Thai
Filing Date:	December 16, 1998	Art Unit:	2186
Title:	Transaction Manager and Cache for Processing Agent		

APPEAL BRIEF

ASSISTANT COMMISSIONER FOR PATENTS
Washington, DC 20231

Sir:

This is an Appeal from an Office Action dated February 26, 2001, finally rejecting each of the pending claims 1-23. The Notice of Appeal was filed on May 29, 2001. The period to file this Appeal Brief expires on December 29, 2001.

REAL PARTY IN INTEREST

The real party in interest in this matter is Intel Corporation. *See* Assignment (recorded February 17, 1999 at Reel 9881, Frame 0608).

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF THE CLAIMS

The application contains claims 1-23 (see Appendix). Claims 1-23 stand rejected under 35 U.S.C. §102(e) as unpatentable over Chittor, et al., U.S. Patent No. 6,061,764 ("Chittor"). This appeal is an appeal from the rejection of all the claims.

STATUS OF THE AMENDMENTS

All amendments to date have been entered.

SUMMARY OF THE INVENTION

An internal memory cache is a memory bank that acts as a temporary storage area bridging an external memory and a processor or other agent. Cache memory is generally faster than external memory. Therefore, a cache memory allows the agent to access data at higher speeds than would normally be possible without the cache. The larger the cache memory, the faster the overall performance of the agent, since there will be a greater chance that a copy of the required instruction or data element will already be in the cache.

Conventionally, agents have exchanged data with one another in cache line-sized increments. The agents' caches and system memory have been organized to have entries of a predetermined width. Whatever the width chosen for these agents, external bus transactions typically have been designed to exchange data in these cache line-sized increments. A "cache line," therefore, has referred not only to the amount of data stored within an entry of an agent's cache, it also has referred to the maximum amount of data that could be exchanged in a single bus transaction. There has been a one-to-one relationship between the cache entries and the maximum amount of data to be transferred in a single bus transaction.

Embodiments of the present invention break this one-to-one relationship as it existed in the prior art. Rather than have an internal cache that is identically sized to the capacity of a single bus transaction, these embodiments provide cache entries that can store several increments of this capacity. To simplify terminology, a "cache line width" refers to the width of individual entries of an agent's cache and a "data line width" refers to the maximum amount of data that can be transferred in a single bus transaction. The inventors are unaware of any prior art that breaks this one-to-one correspondence between the width of a cache entry and the maximum capacity of a transaction on the external bus.

FIG. 2 below illustrates this relationship wherein a single entry 510 stores multiple data line widths 530, 540 of data:

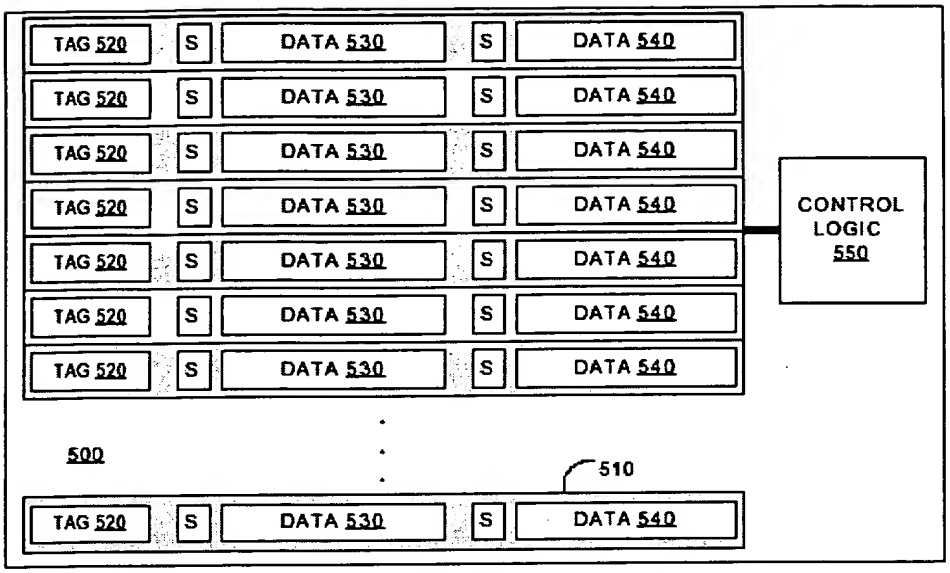


FIG. 2

The entries provide several advantages. First, cache entries 510 may have a single field 520 to store address information for both data lines 530, 540. This architecture may conserve area when the cache is manufactured in an integrated circuit. Additionally, state information S may be stored for each data line 530, 540 permitting cache coherency operations to occur on a data-line basis.

Other embodiments provide for a transaction queue to manage information in the cache. Transaction queues generally manage requests on the external bus. According to the present invention, a single entry of the transaction queue may include fields for multiple transactions on the external bus. FIG 3, reproduced below, illustrates an embodiment of the transaction queue entry:

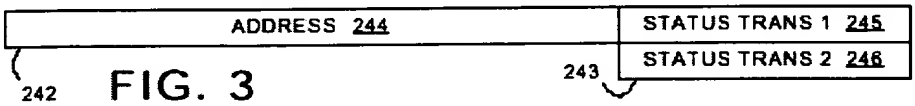


FIG. 3

It may include an address portion 244 to store address information, a first status field 245 to store data associated with the first transaction and a second status field 246 to store data associated with the second transaction. Thus, when a single data request is loaded into the external transaction queue 240, the queue may respond by issuing a sequence of transactions addressed not only to the address of the original data request but also to other addresses proximate thereto.

These embodiments, when used in an agent, provide for a natural prefetch mechanism. Ordinarily, when a data request from the agent's core misses the internal cache, the request propagates to the external bus and a data line-sized unit of data will be supplied to the core and stored in the cache. These embodiments not only retrieve the data line requested by the core, they also retrieve multiple data lines from other locations nearby. Oftentimes, since program flow progresses linearly in the absence of discontinuities, the prefetched data is likely to be used by the core. This prefetch mechanism increases the likelihood that the cache will store data to be used by the core before the core asks for it and, therefore, can increase the agent's performance.

The Examiner rejected all pending claims as anticipated by Chittor and has taken official notice of several features of the pending claims, citing ultimately to Handy, The Cache Memory Book, pp. 78-81 in support thereof. Neither Chittor nor Handy disclose breaking the one-for-one relationship between cache lines and data lines described above. Chittor does not describe any architecture for caches or transaction queues. Therefore, as described more particularly below, the Examiner's cited art is insufficient to anticipate the claimed inventions.

ISSUES

The issues on appeal are whether claims 1-23 patentably distinguish over Chittor under 35 U.S.C. § 102(e).

GROUPING OF CLAIMS

Claims 1-7 are directed to a processing agent with an internal cache having cache entries that are sized to store multiple data line lengths of data.

Claims 8-10 and 22 are directed to an agent transaction queue having a separate status field for each transaction associated with a queue entry.

Claims 11-16 describe a processing agent comprising an internal cache with cache entries sized to store multiple data lines, and a transaction queue system that posts external transactions related to a single data line.

Claims 17-21 and 23 are directed to methods of processing a data request comprising posting a series of external data line transactions for a cache line when the data request misses the cache.

A separate basis for patentability exists for each group of claims. However, except to the extent otherwise indicated below, the respective groups of claims do not stand or fall together for purposes of this appeal.

ARGUMENT

The Examiner rejected claims 1-23 under 35 U.S.C. § 102(e) as anticipated by Chittor. Because Chittor does not disclose each and every element of the Appellants' pending claims, the Examiner's rejection must be reversed.

Chittor describes a memory controller that bridges between a pipelined bus and a serial bus in a computer system. As described therein, a pipelined bus 60 is characterized by a maximum length of data that may be transferred in a single bus transaction. Chittor, Col. 2:34-50. Entries of a system memory are synchronized to this length. A serial bus 70 has no such limitation. It supports data transfers of variable lengths -- lengths that typically are greater than the lengths of pipelined bus transactions. Chittor, Col. 2:10-29. Chittor's primary emphasis relates to resolving cache coherency issues that arise when a request received on the serial bus 70 spans multiple cache lines on the pipelined bus 60. His solution is to issue a sequence of atomic transactions on the pipelined bus 60 for cache coherency purposes before the memory controller supplies data responsive to the serial bus request. Chittor, Col. 7:7-9:43. Chittor nowhere discusses the length of any cache in his system or the architecture of any transaction queue in his system. Chittor does not even suggest to break the one-to-one correspondence between cache entries and the capacity limits of bus transactions.

Claims 1-7 are Allowable Over Chittor.

With regard to claim 1 of the present invention, Chittor fails to teach or suggest a processing agent comprising an internal cache, where *each cache entry is sized to store multiple data line lengths of data*. A data line length is explained quite clearly to be the maximum length of data that may be transferred in a single bus transaction. *See* Application at 1-2. The Examiner rejected claim 1, arguing that Chittor's cache 14 of agent 10 stores multiple data line lengths of data. *See* final Office Action (paper no. 9), p. 3-4; *see also* Chittor, Fig. 1. This is not correct. Of Chittor's entire disclosure, there are only two sentences that discuss the cache 14. They read:

> The MIOC cooperates with other agents to exchange data between the memory 200 and data caches of the agents, such as cache 14. *See* Col. 1:34-36.

> An agent 10 may modify data at an address and store the modified data in an internal cache 14 without notifying other agents on the pipelined bus 60. *See* Col. 2:56-58.

Chittor never discusses the size or architecture of entries within his cache 14. He certainly does not relate the size of these entries to a data line length -- the maximum amount of a data that can be transferred in a single bus transaction. Thus, the Examiner's rejection should be reversed.

Applicants identified these distinguishing features in response to the first Office Action. In the final rejection, however, the Examiner responded that this subject matter is inherently present:

> [T]he cache lines of Chittor has to include [a] multiple of entries that [are] sized to store multiple data line lengths of data; since [the] Chittor invention [is directed] to *variable length read requests* from the cache lines wherein the size of a cache line typically relates to the largest increment of data that may be transferred in a *single pipelined bus transaction.*

See final Office Action (paper no. 9), p. 8 (emphasis added). This argument is incorrect on many levels.

The Examiner's inherency argument is based on an incorrect understanding of Chittor. Chittor discloses an agent 50 (Fig. 1) that communicates to a first set of agents 10-40 via a pipelined bus 60 and to other agents 80 via serial buses 70. A pipelined bus 60 is characterized by a maximum length of data that may be transferred in a single bus transaction. Chittor, Col. 2:34-50. A serial bus 70 has no such limitation. It supports data transfers of variable lengths -- lengths that typically are greater than the lengths of pipelined bus transactions. Chittor, Col. 2:10-29.

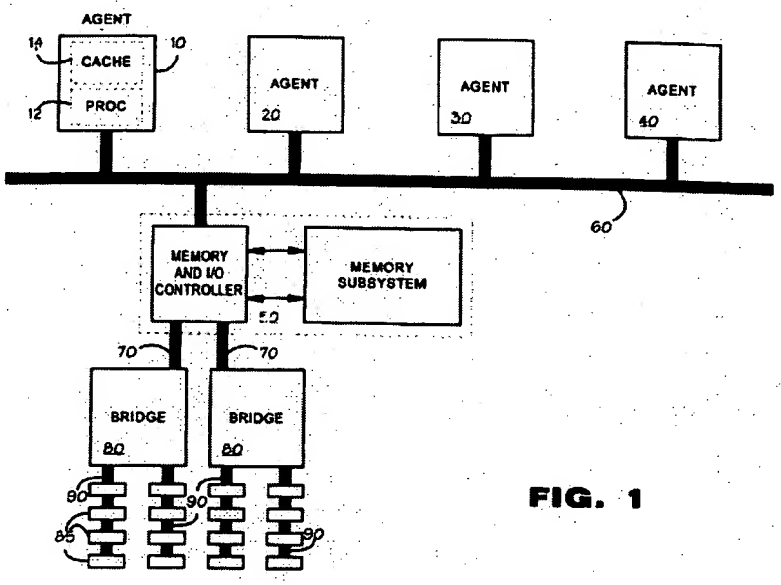


FIG. 1

Chittor, FIG. 1

The Examiner's belief that the pipelined bus 60 supports variable length transactions is incorrect. The *serial bus* 70 supports variable length transactions, not the pipelined bus 60. Chittor clearly states that the pipelined bus 60 can carry a fixed maximum amount of data in a single transaction. When processing a read request from the serial bus 70 (a request that spans multiple cache lines), the agent 50 must issue several "atomic" transactions on the pipeline bus 60 for cache coherency purposes. *See*, Chittor, Col. 7:7-8:67. The agent 50 essentially translates between the protocols of the two busses 60, 70 to which it is connected. Thus, an essential premise of the Examiner's inherency argument, that the cache 14 must support variable length transactions, is incorrect.

The Examiner's inherency argument is flawed for a second reason: Chittor identifies no limit to the length of data transferred in serial bus transactions. Nevertheless, the Examiner thinks that an agent that supports variable length transactions must have a cache that holds multiple lengths of this undefined but varying length of data. This defies logic. Chittor's disclosure does not anticipate the claimed subject matter.

Finally, the Examiner's inherency argument is contradicted by Chittor itself. The Examiner essentially believes that an agent that supports variable length transactions must have cache entries that are sized to be multiple lengths of a data line. The agent 50 supports variable length transactions and has a queue 160 (FIG. 2) to store data for them. Interestingly, however, the width of the queue entries is disclosed to be the *width of a cache line* -- the maximum amount of data transferred in a pipelined bus transaction. Rather than synchronizing the width of these queue entries to any characteristic of the serial bus 70, they are synchronized to the pipelined bus 60 instead. Significantly, their width is *identical* to the maximum amount of data that may be carried on the pipelined bus in a single transaction. This is exactly what the inventors propose to change with the cache structure of claim 1. Again, the Examiner's inherency argument is wrong. Claim 1 is not anticipated by Chittor. For at least the same reasons, claims 2-7, which depend from claim 1, are also not anticipated by Chittor. The Examiner's rejection of claims 1-7 should be reversed.

Dependent claim 4 recites additional features that are not anticipated by Chittor. Claim 4 describes a cache entry having multiple entries, each of which is sized to store a data line length of data, and each of which is associated with its own state field. Claim 4 further recites that a separate cache coherency state field may be employed for each data line portion of the cache entry. This feature permits state information to be maintained for each of several different data line transactions within a single cache entry. The Examiner rejected claim 4 on the basis that Chittor

discloses multiple transactions that span more than one cache line. *See* final Office Action (paper no. 9), p. 4-5; *see also* Chittor Col. 4:5-19, Col. 5:1-9. This discussion, however, implies no structure to the cache itself, nor does it recognize the difference between Chittor's series of transactions that span more than one cache line and the present invention's support of multiple data line length transactions within a single cache entry. Chittor fails to disclose either of the features recited in claim 4. The Examiner's rejection of claim 4 should be reversed. Claims 6-7, which depend from claim 4, also recite these features. The § 102(e) rejections of these claims should be reversed as well.

Claims 11-16 Are Allowable Over Chittor.

Claim 11 recites an architecture for a processing agent that comprises an internal cache having cache entries, each sized to store multiple data lines, and a transaction queue system to post external transactions related to a single data line. The claim also recites that the transaction queue system and the cache each receive data requests on a common input. Chittor discloses none of this subject matter. As noted above, no cache within any of Chittor's agents is disclosed as having a width that is sized to store multiple data lines. This is reason enough to reverse the anticipation rejection to claim 11. Additionally, however, none of Chittor's agents is disclosed as having a cache and a transaction queue that receives data requests via a common input. This is a second basis on which to reverse the anticipation rejection to claim 11.

The Examiner rejected claim 11 on the ground that Chittor cache 14 (FIG.1) and splitter 140 (FIG. 2) show signal lines connecting both a cache and a transaction queue system to a common bus. This interpretation is misleading. First, claim 11 states that both the cache and the transaction queue are in the same agent, but the Examiner has pointed to elements from different agents 10, 50 of Chittor's system. Second, the claim states that both the cache and the transaction queue receive data requests on a common input. Chittor says nothing about where the data requests to the cache 14 originate; presumably the requests originate from within the agent 10 itself. With respect to the splitter 140, however, Chittor clearly discloses that its inputs originate from the inbound transaction queue 130. And, of course, Chittor does not teach that a cache may store multiple data lines while an external transaction relates to a single data line. Chittor does not anticipate the features of claim 11.

Claims 12-16 depend from claim 11. For at least the reasons advanced above, Chittor does not anticipate claims 11-16. The Examiner's rejection of these claims should be reversed.

Claims 8-10 and 22 are Allowable Over Chittor.

Claims 8-10 and 22 define an architecture for an agent *transaction queue that stores multiple transactions per queue entry*. Chittor does not disclose all the elements of these claims and, therefore, cannot anticipate them. Specifically, claim 8 discloses that each transaction queue entry may include two components, each of which defines a separate transaction. A primary component describes a first transaction; it has an address portion and a status portion. A secondary component describes a second transaction; it has a status portion. Claim 22 describes similar subject matter, describing an address portion as a separate field rather than part of the primary component. Chittor does not disclose any transaction queue system having components that describe more than one transaction per queue entry. Thus, the Examiner's rejection must be reversed.

To support the rejection of claims 8-10 and 22, the Examiner cites to Chittor's disclosure of a splitter 140 and bus request generator 122. *See* final Office Action (paper no. 9), p. 5-6; *see also* Chittor Col. 5:1-9, 5:11-24. Chittor's description, however, is not so detailed as to describe the architecture of either the splitter 140 or the bus request generator 122. Chittor does not describe, for example, whether his architecture supports storage of a sequence of transactions in multiple entries or in multiple fields of a single entry. From Chittor's silence, the Examiner conveniently assumes *without supporting disclosure* that the architecture of the splitter 140 and bus request generator 122 are the same as recited in the instant claims. This is impermissible. Chittor must disclose the structure of these claims before he can anticipate them. Because he does not, because the Examiner merely speculates that anticipatory structure is present, the Examiner's rejection of claims 8-10 and 22 under 35 U.S.C. § 102(e) should be reversed.

Claims 17-21 and 23 are Allowable Over Chittor, With or Without *Handy*.

Claim 17 recites a cache management method wherein, in response to a cache miss, *a sequence of external transactions are posted to fill a cache line with data associated with the data request*. Chittor says nothing about data requests and nothing about any agent response to a cache miss. Furthermore, Chittor fails to disclose that an agent may issue a sequence of external transactions to fill a cache line in response to a cache miss. Chittor has no disclosure corresponding to the claimed subject matter of claim 17 and, therefore, claim 17 is allowable over Chittor. The Examiner's rejection of claim 17 must be reversed.

When rejecting claim 17, the Examiner argued that the claim was merely an expression of the subject matter of earlier-rejected apparatus claims. He provided no new analysis of the claim. In the final rejection, in an apparent concession to Chittor's inadequacy as an anticipatory reference, the Examiner took "official notice" that the elements of claim 17 are notoriously well known in the art. *See* final Office Action (paper no. 9), p. 9; *see also*, Advisory Action (paper no. 12), p. 2-3. Applicants disagree. When pressed to justify the official notice, the Examiner cited to Handy, The Cache Memory Book, pp. 78-81.

Claim 17 provides a specific response to a cache miss -- a sequence of external bus transactions that are posted to fill a cache line. While the Examiner may be correct that detection of cache misses and external transactions are known generally, the specific response recited in claim 17 is not. Chittor does not disclose this subject matter. Chittor does not discuss even the general process by which an agent retrieves data in response to a cache miss. Although asked to do so, the Examiner could not cite to a single provision within Chittor that describes this subject matter. *See* Amendment After Final Rejection (filed March 29, 2001), p. 8.

Handy also fails to disclose this subject matter. In § 2.2.6, the Handy reference discusses the use of write buffers during reads and writes of data. As described by Handy, a write buffer stores data evicted from a cache to make room for new data to be read into the cache. Handy explains that, without the write buffer, a write cycle would have to occur before a read of data. With a write buffer, data may be evicted to the write buffer but remain within a processor and new data may be read to the cache. The write of data from the write buffer to main memory may occur at a more leisurely pace. Handy does not describe any system, however, that in response to a cache miss fills a cache line with multiple external transactions.

Claim 17 stands rejected based solely on the Examiner's assertion that it is unpatentable. Chittor does not anticipate the claim and the Examiner's official notice is proven to be false. Because no cited reference discloses the subject matter of claim 17, the § 102 rejection must be reversed.

Claim 23 recites subject matter that is similar to claim 17. When a data request misses an internal cache, claim 23 recites that a series of external transactions are posted to fill a cache line with data associated with the data request. The external transactions are directed to a data-line-sized data item identified by an address of the data request and to at least one other data-line-sized data item adjacent to the first data item. This claim recites not only that a series of data transactions fill a single cache line but also that each of the transactions are directed to a data-line sized data

item. As discussed, Chittor does not disclose this subject matter and the Examiner's official notice is improperly taken. The rejection of claim 23 should be reversed.

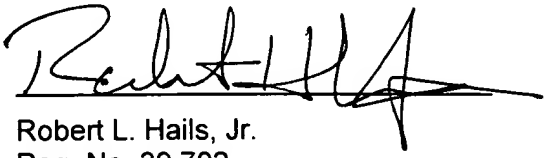
SUMMARY

In view of the above, the Appellants respectfully submit that all claims on appeal distinguish over the cited references and respectfully request that the Examiner's rejections of these claims be reversed. The Appellants therefore respectfully move this Board to reverse the Examiner's decision rejecting claims 1-23.

The Commissioner is hereby authorized to charge the appeal brief fee of $310.00 and any additional fees which may be necessary for consideration of this Appeal to Kenyon & Kenyon Deposit Account No. 11-0600.

Respectfully submitted,

Date: July 30, 2001



Robert L. Hails, Jr.
Reg. No. 39,702

KENYON & KENYON
1500 K Street, NW
Washington, DC 20005
Phone: (202) 220-4200
Facsimile: (202) 220-4201

Appendix: Claims on Appeal

(Brief of Appellants Prudvi, et al.,
U.S. Patent Application Serial No. 09/212,291)

1. A processing agent to transfer data of a predetermined data line length in an external transaction, the agent comprising an internal cache having a plurality of cache entries, each entry sized to store multiple data line lengths of data.

2. The processing agent of claim 1, wherein the cache entries include a tag portion adapted to store address information.

3. The processing agent of claim 2, wherein the internal cache further comprises
match detection logic for the tag portions, and
control logic provided in communication with the match detection logic.

4. The processing agent of claim 1, wherein the cache entries include a cache coherency state field in association with each data line length of data.

5. The agent of claim 1, further comprising a transaction queue having a plurality of queue entries, the queue entries including a primary entry adapted to store address information and status information of a first external transaction and a secondary entry adapted to store status information of a second external transaction.

6. The agent of claim 4, wherein the status information of the first external transaction includes a field representing whether the first external transaction is part of a multiple transaction sequence.

7. The agent of claim 4, wherein the total number of primary and secondary entries equals the multiple number of data line lengths provided in the cache entries.

8. A processing agent, comprising a transaction queue having a plurality of a queue entries, the queue entries each further comprising:
a primary sub entry including an address portion and status portion, the status portion provided for a first external transaction of the agent, and

a secondary sub entry including a status portion provided for a second external transaction.

9. The transaction queue of claim 8, wherein the status portion of the primary entry includes a field representing whether the first transaction is part of a multiple transaction sequence.

10. The transaction queue of claim 8, further comprising control logic adapted to cycle through the queue entries and post transactions therefrom.

11. A processing agent, comprising:

an internal cache having cache entries each sized to store multiple data lines, and

a transaction queue system to post external transactions, each external transaction related to a single data line,

wherein the internal cache and the transaction queue system each receive data requests on a common input.

12. The processing agent of claim 11, wherein the internal cache and the transaction queue system communicate by signal lines.

13. The processing agent of claim 12, wherein the signals lines include a cache hit signal line and a tag hit signal line.

14. The processing agent of claim 11, wherein the transaction queue system comprises a plurality of queue entries, each queue entry comprising:

a primary entry including an address portion and status portion, the status portion provided for a first external transaction of the agent, and

a secondary entry including a status portion provided for a second external transaction.

15. The transaction queue of claim 14, wherein the status portion of the primary entry includes a field representing whether the first transaction is part of a multiple transaction sequence.

16. The transaction queue of claim 14, further comprising control logic adapted to cycle through the queue entries and post transactions therefrom.

17. A method of processing a data request within a processing agent comprising:

posting the data request internally within the agent,

determining whether the request hit the cache,

when the request misses the cache, posting a sequence of external transactions to fill a cache line with data associated with the data request.

18. The method of claim 17, wherein the determining step includes:

comparing address information of the data request with tags stored in the internal cache, and

identifying a cache miss when the address information does not match any stored tag.

19. The method of claim 18, wherein the determining step further includes:

when address information matches a stored tag, reading cache coherency state information associated with the requested data, and

identifying a cache miss when the cache coherency state information is invalid for a request type of the data request.

20. The method of claim 17, further comprising, when the request hits the cache:

determining whether the request hits a tag stored in the cache, and

if so, generating a single external transaction to read the requested data into the agent.

21. The method of claim 20, wherein the second determining step includes:

comparing address information of the data request with tags stored in the internal cache, and

identifying a tag hit when the address information matches a stored tag.

22. An agent comprising a transaction queue, the transaction queue further comprising a plurality of queue entries, each queue entry comprising:

an address field,

a first status field to store data associated with a first transaction based on the address field, and

another status field to store data associated with another transaction based on the address field.

23. A method of processing a data request within a processing agent comprising:

posting the data request internally within the agent,

determining whether the request hit the cache,

when the request misses the cache, posting a series of external transactions to fill a cache line with data associated with the data request, the external transactions directed to a data-line-sized data item identified by an address of the data request and to at least one other data-line-sized data item adjacent to the first data item.